Data generator with a control over linear separability

Nejc Ilc

University of Ljubljana, Faculty of computer and information science
E-mail: nejc.ilc@fri.uni-lj.si

Abstract

Machine learning algorithms are developed to discover hidden structure in data, e.g. algorithms for classification and clustering. Usually, they address specific characteristics of data to optimize their performance. Assessment and comparison of algorithms require enough data, which demonstrate different levels of complexity of certain characteristic under investigation. One example of challenging characteristic is a linear separability of classes. We propose a generator of two-dimensional data that produces non-overlapping set of points of various shapes and can control the level of linear separability between classes. Thus, it enables benchmarking of algorithms in a controllable and systematic manner.

1 Introduction

When assessing the performance of machine learning algorithms, we usually measure their abilities on a collection of datasets from a certain problem domain. If these data come from the real world measurements, we cannot manipulate with the intrinsic characteristics of data. However, this would be essentially useful when benchmarking specific features of learning algorithms. This is the reason some researchers create their own data in a controllable manner. We call this procedure a synthetic or artificial data generation.

One of the main challenges in machine learning, as stated by the authors of the Fundamental Clustering Problems Suite [1], are data with classes that cannot be separable by a hyperplane – we say they are not linearly separable. In other words, data points of one class lie partially or completely inside the convex hull of points from other class. Recently, Elizondo et al. [2] even proposed to measure the level of data complexity by means of linear separability. To the best of our knowledge, no data generator has been devised so far that is able to control the amount of linear separability between classes. Our contribution, a data generator, fills this void and opens the possibility to systematically study and compare the performance of learning algorithms on data with controllable distances and amount of linear separability between classes.

The remainder of the paper is organised as follows. In Section 2, we give an overview of existing synthetic data generators. We introduce the proposed data generator in the Section 3 and demonstrate its capabilities in Section 4. Finally, Section 5 concludes the paper with some future work directions.

2 Related work

Little of published research in the field of data mining had addressed systematic and reproducible synthetic data generation before the work of Pei and Zaïane [3] in 2006, who developed a versatile data generator for the purpose of assessing the algorithms for clustering and outlier detection. Their generator outputs two-dimensional numeric data and enables a user to control the number and density of data points in each class, the number of classes in dataset, the shape of point-sets classified into five difficulty levels, the density function, and the level of background noise. The generator allows us to set the distance between points in different classes indirectly by adjusting shapes' difficulty and density levels of classes. Consequently, the classes in the generated dataset often overlap, as we cannot define a minimal gap between the nearest data points in the adjacent classes. We illustrate this issue in Fig. 1, showing a screen-shot of Pei and Zaïane data generator user interface, where five overlapping classes were generated.

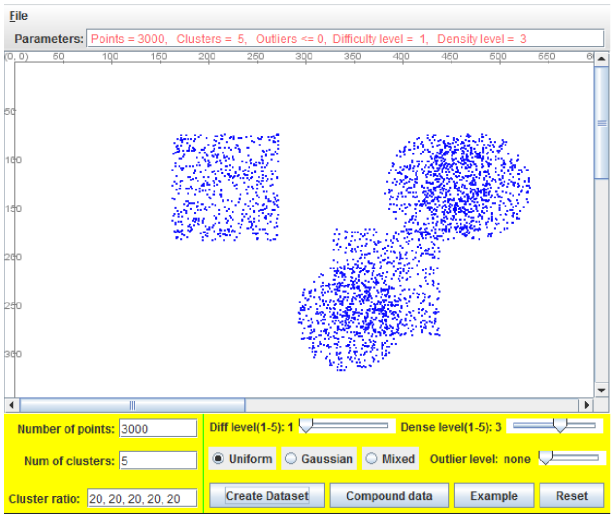

Figure 1: Problem of overlapping classes when using Pei and Zaïane data generator.

When the purpose of a data generator is to mimic or reproduce real-world problems, one may find it beneficial to exploit a model of real data, which is what Eno and Thompson did [4]. They proposed to build a model of input data with decision trees and describe both the data and the model with Predictive Model Mark-up Language (PMML). Their main contribution is a generation of synthetic data of any size with similar patterns as in the original data. Analogous concept is investigated by Marko Robnik-Šikonja [5] who proposed a semiartificial data generator based on radial basis function neural networks.

Adä and Berthold developed the Modular Data Generator – an extension module of a visual data-mining tool KNIME [6]. This module can be incorporated into a data-processing work-flow and can be extended with new processing modules. It is quite universal and covers data generation for various problem domains, like shopping-basket analysis, association rules, and cluster analysis. The shortcoming of this generator is that we cannot control the distance between the classes dynamically as their positions have to be predefined.

Frasch et al. proposed a data generator with controllable statistical properties [7] – the emphasis is on the Bayes error rate. Data are generated by white Gaussian densities with their means on the corner of a regular k-simplex, e.g. vertices of a triangle in a 2-dimensional space. We found it rather inappropriate due to the generation of the overlapping classes. Also, they are at most hyper-ellipsoidal and therefore of a moderate difficulty level.

3 Data generator with a control over linear separability

We developed a synthetic data generator that is able to produce two-dimensional datasets with non-overlapping sets of points. A user can specify the number of classes, the number of data points in each class, the shape and distribution of data points in a class, the minimal distance between classes, and the degree of classes that are linearly separable (LS), i.e. how many pairs of classes are separable by a line or a hyperplane in general.

To test whether two classes of data points are LS, we employ linear programming using the Simplex method [8]. We are searching for a hyperplane H in $\mathbb{R}^D$ such that it separates data points in one class from another. If we find such a hyperplane, the corresponding classes are LS and if not, we say they are not LS.

The data generator's code is written in MATLAB and is available at `https://github.com/nejci/data-generator-lin-sep`.

3.1 Shape of a point-set

In order to manipulate distance and linear separability between classes we have to define the "body" of points in a class. Here we employ α-shapes [9, 10] as a tool for describing the shape or silhouette of a set of points (point-set) in the plane. The α-shape is a generalization of the convex hull parametrized by the real non-negative number α that controls the level of details of the shape. The smaller α is, the more detailed the shape of a point-set gets, meaning that the boundary fits tighter around the point-set, and vice-versa. So, when $\alpha \to \infty$ the α-shape becomes the convex hull of the data points in the class. We define an α-shape as follows: if the bounding circle of an empty open disk with radius α passes through two points then there is an edge connecting these two points; α-shape is a set of all the edges that satisfies this condition. We set the value for α as the smallest disk radius that produces an α-shape with only one region. Furthermore, to avoid fragmented shapes with small holes in the middle we define an area threshold under which the holes are suppressed. In our experiments, the 1% of a bounding-box area suffice. In Fig. 2 we depict a class that consists of 300 data points sampled from a shape of the letter 'A' and the α-shape of this point-set. A close-up view demonstrates the construction of an α-shape – there are two circles with the radius α that intersects with exactly two boundary points.

Our generator can produce 38 different shapes, some of them are depicted in Fig. 3. The majority of the shapes are identical to those defined by Pei and Zaïane [3] and demonstrate different levels of complexity: compact and spherical; elongated; with corners and holes that enable embedding of points from other classes. Class members can be drawn from the uniform or truncated normal distribution; the user can choose from the two or allow the algorithm to pick one at random, i.e. mixed distribution.

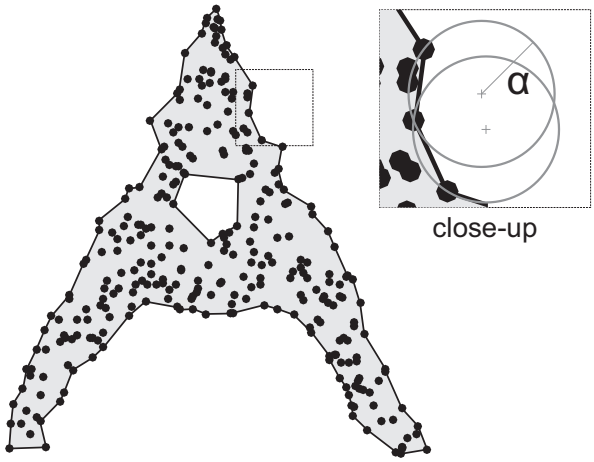

Figure 2: An example of an α-shape on a point-set. Black dots represent data points in a class, gray patch represents the body of a point-set, and a black outline is a α-shape's boundary.

3.2 Data generation algorithm

The main feature of our proposal is a control of the distance and linear separability between classes. Our generator produces two-dimensional data with non-overlapping, crisp classes. This enables simple and efficient visual inspection. We developed an iterative algorithm that tries to achieve this in a two-step procedure: first, we create a

new point-set and compute its α-shape. Then, it is being moved towards the existing shapes until it collides with one of them; second, fine-tuning of the new point-set's position is done by random rotations and translations. The proposed algorithm and its inputs are discussed in greater details in the following.

Input parameters:

- K: number of classes to generate,
- $\mathbf{N}$: number of data points in each class; a vector of K positive integers $\mathbf{N} = \{n_1, n_2, \ldots, n_K\}$,
- d_{min}: minimal distance between every pair of classes,
- L: number of pairs of classes that are not LS.

Additional algorithm options:

- S: *stiffness* - how many percent of distance between selected pair of classes is reduced on each iteration on a coarse level [default: 0.5],
- I_{coarse}: the number of iterations in the first step, i.e. coarse level of movement [default: 300],
- I_{fine}: the number of fine-tuning iterations in the second step [default: 200],
- β_{coarse}: maximum angle of a random rotation in the first step [default: π],
- β_{fine}: maximum angle of a random rotation in the second step [default: $\frac{\pi}{2}$],
- `tol`: tolerance of d_{min} [default: 10% of d_{min}],
- `shapes`: list of shapes,
- `dist`: distribution of data points in the class; can be *uniform*, *truncated normal*, or *mixed*, i.e. the distribution is chosen at random between uniform and truncated normal for each class [default: uniform],
- r_{gen}: a distance from the centroid of the first-created point-set to the position of the newly generated set [default: 20].

A pseudocode of the data generator is listed in Algorithm 2, helper functions are defined in Algorithm 1.

4 Evaluation and discussion

For illustration, we generated four datasets and plotted them in Fig. 3 using various input parameters' values of the generator. The total number of data points in each dataset is 500. The datasets were scaled proportionally to fit in the interval $[0, 1]$ in both dimensions. The main generator's parameters are listed in Table 1. We used the default options' values that are listed in Section 3.2.

We have already used the data generator in our thesis [11] to produce benchmark datasets for the performance evaluation of algorithms for cluster analysis and validation. Considering the adjusted Rand index for measuring the similarity between a clustering and the ground truth, the algorithms performed less accurate when we increased the number of classes that are not LS or we decreased the distances between classes.

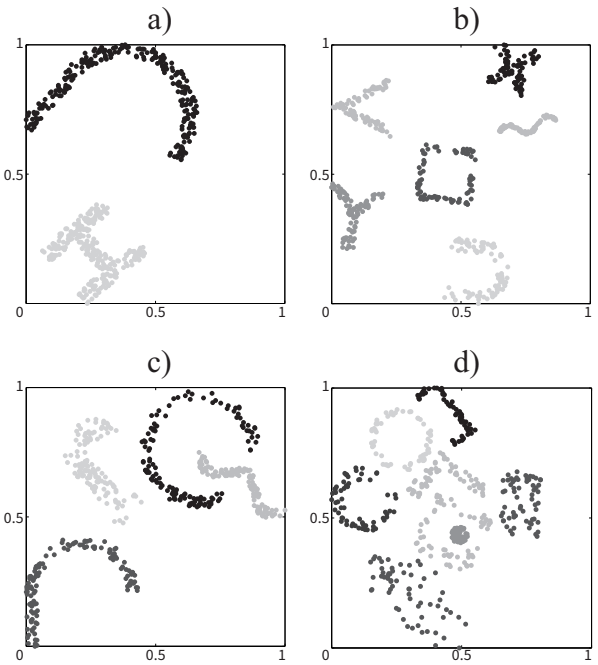

Figure 3: Example datasets.

Table 1: Generator's parameters used for examples.

Example	K	d_{min}	L	`dist`
a)	2	0.50	0	uniform
b)	6	0.30	0	uniform
c)	4	0.10	1	trunc. norm.
d)	8	0.08	2	mixed

5 Conclusion

We presented a novel synthetic generator of two-dimensional data with a control over the linear-separability of classes. It is capable of creating point-sets of complex shapes with controllable minimal distance and the degree of linear-separability between classes. The generator's code is open-source and we hope it will be helpful for creating benchmark datasets for the systematic and controllable evaluation of machine learning algorithms.

We plan to extend the presented data generator in the future to produce datasets of dimensionality higher than two. We will have to implement algorithm for n-dimensional α-shapes as described in [12] and adapt the routines for scaling, translation, and rotation of a point-set.

References

[1] A. Ultsch, "Clustering with SOM: U*C," in *Workshop on Self-Organizing Maps (WSOM 2005)*, (Paris, France), pp. 75–82, 2005.

[2] D. A. Elizondo, R. Birkenhead, M. Gamez, N. Garcia, and E. Alfaro, "Linear separability and classification complexity," *Expert Systems with Applications*, vol. 39, no. 9, pp. 7796–7807, 2012.

[3] Y. Pei and O. Zaïane, "A synthetic data generator for clustering and outlier analysis," tech. rep., Department of Computing science, University of Alberta, 2006.

[4] J. Eno and C. W. Thompson, "Generating synthetic data to match data mining patterns," *IEEE Internet Computing*, vol. 12, pp. 78–82, 2008.

[5] M. Robnik-Šikonja, "Data Generators for Learning Systems Based on RBF Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 5, pp. 926–938, 2016.

[6] I. Adä and M. R. Berthold, "The New Iris Data: Modular Data Generators," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 413–422, 2010.

[7] J. V. Frasch, A. Lodwich, F. Shafait, and T. M. Breuel, "A Bayes-true data generator for evaluation of supervised and unsupervised learning methods," *Pattern Recognition Letters*, vol. 32, no. 11, pp. 1523–1531, 2011.

[8] D. Elizondo, "The linear separability problem: Some testing methods," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 330–344, 2006.

[9] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 1983.

[10] H. Edelsbrunner, "Alpha Shapes – a Survey," *Tessellations in the Sciences*, pp. 1–25, 2010.

[11] N. Ilc, *Clustering Based on Weighted Ensemble*. PhD thesis, University of Ljubljana, 2016.

[12] H. Edelsbrunner, "Weighted alpha shapes," tech. rep., University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

Algorithm 1 Helper functions

function POINTSETCREATE(N)
  `shape` ← random element from the list `shapes`
  $\mathbf{c}$ ← generate N data points that fill the `shape` using the distribution `dist`.
  Scale points in $\mathbf{c}$ by a random factor on $[0, 1]$.
  return $\mathbf{c}$
end function
function POINTSETMOVE($\mathbf{c}, \mathbf{a}, \mathbf{b}$)
  Move every point in the set $\mathbf{c}$ by $\mathbf{b} - \mathbf{a}$.
  return $\mathbf{c}$
end function
function POINTSETROTATE($\mathbf{c}, \mathbf{a}, \beta_{\text{max}}$)
  β ← random angle in the range $[0, \beta_{\text{max}}]$.
  Rotate points in $\mathbf{c}$ around point $\mathbf{a}$ by the angle β.
  return $\mathbf{c}$
end function
function BND($\mathbf{c}$)
  $\mathbf{b}$ ← points on a α-shape boundary of the set $\mathbf{c}$
  return $\mathbf{b}$
end function

Algorithm 2 Data generator

Input: $\mathbf{N}$, K, d_{min}, L
Output: $\mathbf{X}$

$\mathbf{c}_1 \leftarrow$ POINTSETCREATE(n_1)
$\mathbf{c}_1 \leftarrow$ POINTSETROTATE($\mathbf{c}_1, \overline{\mathbf{c}_1}, 2\pi$)
$\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{c}_1$
for $i \leftarrow 2$ to K
  $\mathbf{c}_i \leftarrow$ POINTSETCREATE(n_i) ▷ $n_i \in \mathbf{N}$
  $\mathbf{c}_i \leftarrow$ POINTSETROTATE($\mathbf{c}_i, \overline{\mathbf{c}_i}, 2\pi$)
  $\mathbf{b} \leftarrow$ random point on a circle with radius r_{gen}
  $\mathbf{c}_i \leftarrow$ POINTSETSMOVE($\mathbf{c}_i, \overline{\mathbf{c}_i}, \mathbf{b}$)
  $\mathbf{c}_i^* \leftarrow \mathbf{c}_i$
  for iter$_c \leftarrow 1$ to I_{coarse} ▷ Step 1: coarse moves
    $\mathbf{c}_{\text{fix}} \leftarrow$ random point-set among 1 to $(i-1)$
    $\mathbf{m} \leftarrow \arg\min_{\mathbf{x}}\{\min_{\mathbf{y}} d_E(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x} \in$ BND($\mathbf{c}_i$) and $\mathbf{y} \in$ BND($\mathbf{c}_{\text{fix}}$)
    $\mathbf{m}' \leftarrow \mathbf{m} + S \cdot (\overline{\mathbf{c}}_{\text{fix}} - \mathbf{m})$
    $\mathbf{c}_i \leftarrow$ POINTSETMOVE($\mathbf{c}_i, \mathbf{m}, \mathbf{m}'$)
    $\beta \leftarrow \text{rnd}(-1, 1) \cdot \beta_{\text{coarse}} \cdot d_E(\mathbf{m}', \overline{\mathbf{c}}_{\text{fix}}) \,/\, r_{\text{gen}}$
    ▷ rnd$(-1, 1)$ is a random number on $(-1, 1)$
    $\mathbf{c}_i \leftarrow$ POINTSETROTATE($\mathbf{c}_i, \mathbf{m}', \beta$)
    `fineTuned` ← 0
    for iter$_f \leftarrow 1$ to I_{fine} ▷ Step 2: fine-tunning
      $d \leftarrow \min_{\mathbf{x},\mathbf{y}}\{d_E(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x} \in$BND($\mathbf{c}_i$) and $\mathbf{y} \in \bigcup_{k=1}^{i-1}$BND($\mathbf{c}_k$)
      if $d < d_{\text{min}}$ ▷ If too close, move $\mathbf{c}_i$ away.
        $\mathbf{m}, \mathbf{f} \leftarrow \arg\min_{\mathbf{x},\mathbf{y}}\{d_E(\mathbf{x}, \mathbf{y})\}$, where $\mathbf{x}, \mathbf{m} \in$BND($\mathbf{c}_i$) and $\mathbf{y}, \mathbf{f} \in \bigcup_{k=1}^{i-1}$BND($\mathbf{c}_k$)
        $\mathbf{m}' \leftarrow \mathbf{m} + (\mathbf{m} - \mathbf{f}) \,/\, d \cdot |d - d_{\text{min}}|$
        $\mathbf{c}_i \leftarrow$ POINTSETMOVE($\mathbf{c}_i, \mathbf{m}, \mathbf{m}'$)
        $\beta \leftarrow \text{rnd}(-1, 1) \cdot \beta_{\text{fine}} \cdot (1 - \text{iter}_f \,/\, I_{\text{fine}})$
        $\mathbf{c}_i \leftarrow$ POINTSETROTATE($\mathbf{c}_i, \overline{\mathbf{c}}_i, \beta$)
      else
        `fineTuned` ← 1
        break ▷ End of fine-tuning.
      end if
    end for
    $l \leftarrow$ num. of pairs of classes that are not LS
    $l_i \leftarrow$ num. of classes that are not LS with $\mathbf{c}_i$
    if $\mathbf{c}_i$ overlaps with any other class OR NOT `fineTuned` OR $l > L$
      $\mathbf{c}_i \leftarrow \mathbf{c}_i^*$ ▷ Reset move.
    end if
    `stopDist` ← 0
    `stopLS` ← 0
    if `fineTuned` AND $|d - d_{\text{min}}| \leq$ `tol`
      `stopDist` ← 1
    end if
    if $l == L$ OR ($L > 0$ AND $l_i > 0$)
      `stopLS` ← 1
    end if
    if `stopDist` AND `stopLS`
      break ▷ Point set $\mathbf{c}_i$ is in its final position.
    end if
  end for
  $\mathbf{X} \leftarrow \mathbf{X} \cup \mathbf{c}_i$
end for